Top-Down vs Bottom-Up Model-Based Methodologies for Distributed Control: A Comparative Experimental Study

Grégory Mermoud, Utkarsh Upadhyay, William C. Evans, and Alcherio Martinoli

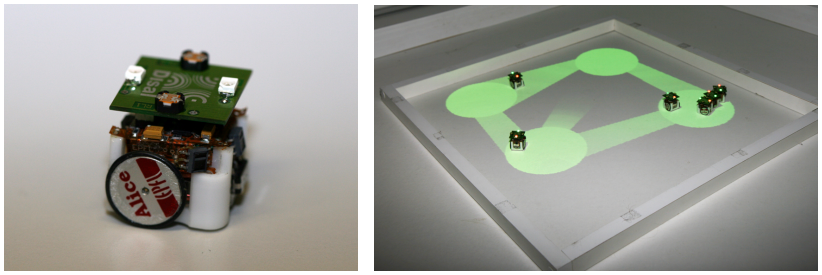
Abstract Model-based synthesis of distributed controllers for multi-robot systems is commonly approached in either a *top-down* or *bottom-up* fashion. In this paper, we investigate the experimental challenges of both approaches, with a special emphasis on resource-constrained miniature robots. We make our comparison through a case study in which a group of 2-cm-sized mobile robots screen the environment for undesirable features, and destroy or neutralize them. First, we solve this problem using a top-down approach that relies on a graph-based representation of the system, allowing for direct optimization using numerical techniques (e.g., linear and non-linear convex optimization) under very unrealistic assumptions (e.g., infinite number of robots, perfect localization, global communication, etc.). We show how one can relax these assumptions in the context of resource-constrained robots, and explain the resulting impact on system performance. Second, we solve the same problem using a bottom-up approach, i.e., we build up computationally efficient and accurate models at multiple abstraction levels, and use them to optimize the robots' controller using evolutionary algorithms. Finally, we outline the differences between the top-down and bottom-up approaches, and experimentally compare their performance.

1 Introduction

Model-based synthesis of distributed controllers for multi-robot systems is commonly approached in either a bottom-up or top-down fashion. In this paper, we propose an experimental comparison of both approaches based on a case study in

Grégory Mermoud, e-mail: gregory.mermoud@epfl.ch
Utkarsh Upadhyay, e-mail: utkarsh.upadhyay@epfl.ch
William C. Evans, e-mail: william.evans@epfl.ch
Alcherio Martinoli, e-mail: alcherio.martinoli@epfl.ch

École Polytechnique Fédérale de Lausanne, School of Architecture, Civil and Environmental Engineering, Distributed Intelligent Systems and Algorithms Laboratory, 1015 Lausanne



(a) (b)

Fig. 1 (a) Close-up picture of an Alice 2002 robot with its extension board. (b) Picture of an on-going top-down experiment with six Alice robots and four spots.

which a group of severely resource-constrained robots screen the environment for undesirable features (e.g., cancer cells, pollutants), and destroy or neutralize them. Such robots are likely to have poor sensing capabilities, and thus may be prone to confusing "good" and "bad" features. Further, they may not be sufficiently powerful or large enough to destroy such features alone.

In a previous contribution, we employed collaboration and consensus between robots as a simple and scalable mechanism for solving this problem using a bottom-up model-based approach [11]. In the present paper, we build upon our previous work to gain deeper insights into the limitations of miniature robots, as well as into the model-based methodologies that have been proposed for the design and control of large groups of such miniature robots.

The design and control methodology that we used in [11] is essentially bottom-up; that is, we start with a real implementation of the real system (or faithful simulations of the system), and then build up a series of increasingly abstract models, carefully validating each against those at lower abstraction levels. This approach has been very successful in designing a large variety of distributed systems [8, 10, 13]. One important benefit of this approach is its direct anchoring to reality, enabling one to predict and optimize the performance of the real system. However, bottom-up approaches generally yield macroscopic models that are difficult to analyze mathematically (e.g., non-linear, time-delayed systems of differential or difference equations, sometimes partial). More critically, while they allow for a precise design of the microscopic behavior based on simple, robust techniques (e.g., behavior-based control), they also require a good deal of intuition for achieving the desired coordinated behavior at the macroscopic level (i.e., they are non-constructive).

In contrast, top-down approaches start with an idealized, often highly abstracted representation of the system (e.g., graph-based), thereby benefiting from the full breadth of related analytical tools, including very powerful design and optimization schemes (e.g., convex optimization, integer programming, linear control methods, etc.). However, this design methodology generally requires very strong and unrealistic assumptions (e.g., perfect localization, discrete environment, absence of noise, etc.), possibly leading to degraded performance when implemented on the target system. Furthermore, top-down approaches usually neither predict real sys-

Table 1 Notation used in the paper. Note that the notation of variable x without mentioning the time (by opposition to the notation $x(t)$) is a simplified way to write $x(\infty)$, i.e., the value of the variable after convergence has been attained.

Variable	Description
N	Number of spots
R	Number of robots
$\mathbf{K} = (k_{ij})$	Transition rate matrix
$f(\cdot)$	Objective function for the Semi-Definite Program (SDP)
$\mathbf{x}^d = (x_i^d)$	Desired distribution of robots
$x_i(t)$	Number of robots at spot i
$1_{i,j}(t)$	1 if robot j is at spot i at time t
$\chi(\kappa, \hat{\kappa})$	Probability that a spot with threshold κ is measured as a spot with threshold $\hat{\kappa}$
Λ_κ	Destruction rate of spots with threshold κ
N_κ	Number of different thresholds a spot can have

tem performance nor provide bounds for performance loss; complete collapse of the collective dynamics is not to be unexpected, especially when dealing with highly stochastic systems, as is often the case with swarms of miniature robots.

Our contribution is two-fold. First, we show how top-down approaches (widely used by other researchers [1, 9]) can be applied to a case study that was specifically designed as a benchmark for control methodologies for resource-constrained distributed robotic systems. In particular, we propose a top-down solution based on graph theory to the problem presented in [11]; this approach yields a *semi-centralized* control, in which robots are essentially autonomous, but exploit information broadcasted by a central planner. Second, we propose a bottom-up solution that relies solely on local information, and yields a fully distributed controller, which is optimized for a wide range of scenarios.

This paper is organized as follows: Section 2 provides a detailed description of our particular case study, and Section 3 describes both the top-down and bottom-up solutions. Section 4 analyzes the performance of our approach using a combination of simulation and real experiments. Section 5 summarizes our findings and discusses future work. The notation used in this paper is summarized in Table 1.

2 Problem Statement

The problem described and solved in [11] is a generalization of the stick pulling experiment [5], in which pairs of robots must collaborate to pull sticks out of the ground. Here, sticks are replaced by N circular patches of light (referred to as *spots*) randomly scattered throughout a bounded arena of area A_{tot}. R robots wander about the arena looking for these spots, and attempt to destroy them. This problem is henceforth referred to as the *collaborative spot-destruction problem*. The spots have a random characteristic threshold $\kappa \in [1, \ldots, N_\kappa]$, and will not be destroyed unless

κ robots are inside the spot[1]. Importantly, the intensity of a spot is a Gaussian random variable with mean I_κ and variance σ_κ^2, which both depend on the characteristic threshold κ. As a result, there is a probability $\chi(\kappa,\hat{\kappa})$ that a spot with threshold κ is measured with threshold $\hat{\kappa}$. Furthermore, spots are not destroyed instantaneously if $x_i \geq \kappa_i$ (where x_i is the number of robots at spot i; see Table 1 for all notations used in this paper), but they rather undergo a destruction process at a constant rate ρ, i.e., they have a "health level" that starts at 1, and decreases at constant rate ρ as long as κ robots (or more) are present in the spot. Upon destruction (i.e., when its health reaches zero), the spot is relocated at a random location in the arena, and the experiment continues. The performance of the system is simply given by the weighted rate of destruction of the spots

$$Perf. = \sum_{\kappa=1}^{N_\kappa} \kappa \cdot \Lambda_\kappa \tag{1}$$

where $\Lambda_\kappa = \rho \sum_{i=1}^{N} \mathbf{1}_{\{x_i \geq \kappa_i \text{ and } \kappa_i = \kappa\}}$ is the destruction rate of spots with threshold κ.

2.1 Experimental Setup

Our comparison relies mainly on a series of experiments with real miniature robots. In this study, we use the Alice 2002 mobile robot [2] (Figure 1(a)), which has a size of 2cm×2cm×2cm, a differential wheel drive that reaches speed up to 4cm/s, and four infrared sensors that allow the detection of passive obstacles at ranges up to 3cm simultaneously with 4bps bidirectional communication up to 6cm. The testing environment is a 50cm square arena. Local infrared communication allows aggregated robots to exchange some bits of information, albeit with poor reliability. In the context of this work, we developed an extension board endowed with two epoxy-encapsulated photo-sensitive sensors (*A9950 11* photocell, Perkin Elmer, maximal spectral sensitivity at 530nm) on each side, which allow the robots to "sense" the gradients and the spots displayed by an overhead projector (see Figure 1(b)).

Obviously, the robots cannot directly interact with these spots, so we employ an overhead camera in conjunction with SwisTrack, an open-source object tracking tool targeted to multi-agent systems [7], to emulate this interaction in software. In order to obtain an accurate measure of both the position and the orientation of the robots, we use markers that consist of two LEDs of different colors (red and green). Our system constantly integrates the trajectory of each robot and updates the environment accordingly, i.e., it detects the destruction of spots and modifies the display by relocating destroyed spots.

[1] Note that, in [11], we made a distinction between "bad" spots that need to be destroyed and "good" spots that must be preserved, but κ was kept constant. In this paper, we assume all spots to be "bad", but with varying thresholds κ_i.

3 Control Strategies

3.1 Preliminaries

The *collaborative spot-destruction problem* is a typical dynamical allocation problem, i.e., one requires the presence of κ_i robots at spot i, and the robots must be re-distributed after each spot destruction as fast as possible. The solution to this problem is two-fold: (1) one needs to determine the optimal distribution of robots $\mathbf{x}^d$ over the set of spots, and (2) one needs a strategy to distribute the robots according to $\mathbf{x}^d$ in a *scalable* and *efficient* manner.

Section 3.1.1 provides a concise solution to the first problem, which depends only on the number of robots R, the number of spots N, and their respective characteristic thresholds κ.

Sections 3.2 and 3.3 describe two distinct model-based strategies (*top-down* and *bottom-up*, respectively) to distribute the robots according to the desired distribution $\mathbf{x}^d$. The relevance and the performance of these strategies depend not only on the number of robots, but also on their capabilities (e.g., computational power, communication and sensing capabilities, localization, etc.) as well as the amount of information available to the central planner, if any.

3.1.1 Optimal desired distribution

Determining the desired distribution $\mathbf{x}^d$ that maximizes the system performance (Equation (1)) is non-trivial, especially if $N_r < \sum_{i \in \mathcal{S}} \kappa_i$ where $\mathcal{S}$ denotes the set of spots. Fortunately, one can construct the desired distribution $\mathbf{x}^d$ such that the performance of the system is optimal[2] by solving the following optimization problem:

$$\begin{aligned} \mathbf{x} &= arg\max_{\mathbf{x}} \; E[\text{Perf.}|\hat{\kappa}] = arg\max_{\mathbf{x}} \; \sum_{\kappa=1}^{N_\kappa} E[\kappa \cdot \Lambda_\kappa|\hat{\kappa}] \\ &= arg\max_{\mathbf{x}} \; \sum_{\kappa=1}^{N_\kappa} \kappa \cdot E[\Lambda_\kappa|\hat{\kappa}] \qquad (2) \\ E[\Lambda_\kappa] &= \rho \sum_{i}^{N} Pr\{x_i \geq \kappa_i \cap \kappa_i = \kappa|\hat{\kappa}_i\} \\ &= \rho \sum_{j}^{N} Pr\{x_i \geq \kappa_i\} \frac{\chi(\kappa, \hat{\kappa}_i)}{\sum_{\lambda=1}^{N_\kappa} \chi(\lambda, \hat{\kappa}_i)} \qquad (3) \end{aligned}$$

where, assuming that x_i is a sequence of i.i.d. binomial random variables,

$$Pr\{x_i \geq j\} = \sum_{l=\kappa}^{R} \binom{R}{l} x_i^l (1 - x_i)^{R-l}$$

[2] Note that we assume in this subsection that the system is optimal *after* convergence, and we do not optimize for the speed of convergence.

and $\hat{\kappa}_i$ is the estimate of the threshold of spot i.

Because of the use of binomial random variables, the problem is non-convex. However, one can solve this problem using nonlinear programming (we used the MATLAB function `fmincon`), with initial conditions given by a simple heuristic that distributes the robots on the spots with large κ_i (but smaller than the total number of robots) first.

3.2 Top-Down Strategy

We use a novel technique developed by Berman et al. [1] that allows one, under certain assumptions, to distribute unlabeled robots over a set of spots in an arena in a scalable and efficient manner. To achieve that, an omniscient central planner[3] needs to (i) construct a graph $\mathcal{G} = (\mathcal{S}, \xi)$ whose vertices are the spots, and whose edges are feasible paths connecting the spots, (ii) compute the optimal transition rate matrix $\mathbf{K} = (k_{ij})$ that allows for the fastest convergence, and (iii) broadcast both $\mathcal{G}$ and $\mathbf{K}$ to the robots.

Interestingly, the spot-destruction problem can be thought as an allocation problem, i.e., it requires the presence of κ_i robots at spot i. Our solution is therefore to steer the swarm of robots to a given desired distribution $\mathbf{x}^d$ using the technique described above; upon the destruction of a spot, the central planner computes new $\mathcal{G}$ and $\mathbf{K}$ and broadcasts them to the robots.

However, beyond this intuitive similarity, many details are left to be worked out. In particular, the many assumptions made in [1] need to be relaxed throughout the process of implementation. For instance, we shall optimize the transition rates while retaining their feasibility on a real robotic system (Section 3.2.1); also, we shall determine the desired distribution $\mathbf{x}^d$ for finite number of robots (Section 3.2.2), as well as a method for constructing both $\mathcal{G}$ and $\mathbf{K}$ such that they can be broadcasted to resource-constrained robots such as the Alice robot (sections 3.2.3 and 3.2.4). Hereafter, we discuss each of these assumptions by proposing either an objective criterion for their validity, or a metric of their impact on the system performance.

3.2.1 Optimizing Transition Rates

The optimal transition rate matrix $\mathbf{K}$ is computed by the central planner, which solves a Semi-Definite Program (SDP) [3] that finds the *transition rates* from one spot to another that ensure the fastest convergence to the desired distribution. Moreover, under the constraint $\mathbf{K}\mathbf{x}^d = 0$, which ensures that the robots will indeed converge to the desired distribution $\mathbf{x}^d$, the transition rates can be made such that, after attaining convergence, the system makes as *few* transitions as possible (so that they are feasible in the context of a real robotic system). The objective function used to limit the transitions after convergence can be formulated in two different ways:

[3] For the sake of scalability, the central planner does not know about the position of each robot. Also, in [1], the central planner assumes that there is an infinite number of robots.

$$f(\mathbf{K}) = \sum_{(i,j)\in\xi} k_{ij} x_i^d \tag{4}$$

$$\text{or, } f(\mathbf{K}) = \max_{(i,j)\in\xi} k_{ij} x_i^d \tag{5}$$

Equation (4) represents the total number of transitions per time unit after attaining equilibrium, while Equation (5) represents the maximum number of transitions per time unit. To solve the SDP problem, we use `CVX`, a MATLAB package for specifying and solving convex programs [4].

3.2.2 Infinite Number of Robots

One fundamental assumption of the method proposed in [1] is that an infinite (or at least very large) number of robots are involved in the system. The finiteness of the population introduces a steady state error that grows as the number of robots decreases; an infinite number of robots would allow us to attain any distribution with an arbitrarily small degree of accuracy. However, since we have a finite number of robots, we expect to see some difference between the desired and actual distribution of robots, even after convergence. This steady state error has to be characterized in order to determine whether the given distribution has converged or not. The *distance* of the actual distribution from the desired distribution is defined as the square of the l^2-norm:

$$\|\mathbf{x}(t) - \mathbf{x}^d\|^2 = \sum_{i=1}^{N} \left(\frac{x_i(t)}{R} - x_i^d \right)^2, \tag{6}$$

which is also the definition used by Berman et al. in [1].

Assume that the system is at equilibrium, and $x_i(t) = \sum_{j=1}^{N} 1_{i,j}(t)$. In this case, each robot can be at spot i with probability x_i^d. Also, each robot moves independently of the others. Hence, variables $1_{i,j}$ are independent for each robot[4]. Therefore, we have (assuming $t \to \infty$):

$$\begin{aligned} E[\text{Dist.}] &= E[\frac{1}{R^2} \sum_{i=1}^{N} (x_i - R x_i^d)^2] \\ &= \frac{\sum_{i=1}^{N} E\left[x_i^2 - 2R x_i^d x_i + (R x_i^d)^2\right]}{R^2} \\ E[x_i^2] &= E[\sum_{l,k} 1_{i,l} \cdot 1_{i,k}] = R \cdot x_i^d + R(R-1) \cdot (x_i^d)^2 \\ \text{Hence,} \quad E[\text{Dist.}] &= \frac{1}{R} \sum_{i=1}^{N} (x_i^d - (x_i^d)^2) = \frac{1 - \sum_{i=1}^{N} (x_i^d)^2}{R} \end{aligned} \tag{7}$$

This value in Equation (7) can be calculated for a given desired distribution, number of spots and number of robots. As a result, it is clear that the performance of the real

[4] Actually, they have to follow the additional constraint $\sum_{i=1}^{N} \sum_{j=1}^{R} 1_{i,j} = R$ to conserve the number of robots. However, they are independent for each robot, which is the property used.

system, given its finiteness, will always be overestimated; Equation (7) is a measure of how inaccurate the actual distribution will be at steady state for a finite population of robots.

3.2.3 Graph Structure

One general assumption made by Berman et al. [1] is that the underlying graph $\mathcal{G}$ is strongly connected (i.e., a directed path exists between any pair of distinct vertices). As discussed in Section 3.2.4, our method of broadcasting the graph structure to the robots imposes that edges must be unidirectional; indeed, gradients must be wide enough for robots to follow them while being non-overlapping .

In this paper, we construct a maximal planar graph using Delaunay triangulation (in $O(n \log n)$ time), and then an arbitrary triangle is turned to a cycle, and is chosen as the starting graph. The other spots are added one by one, keeping the graph strongly connected. Compared to other graph structures such as cycles, Delaunay graphs have maximum connectivity under our constraints and therefore result in the fastest convergence.

3.2.4 Broadcast and Navigation

Another fundamental assumption made by Berman et al. [1] is that the central entity can broadcast the graph structure $\mathcal{G}$ and the matrix $\mathbf{K}$ to the robots. In turn, the robots shall be able to navigate along the edges of the graph. However, resource-constrained robots such as the Alice are generally endowed with very poor and unreliable sensing, communication, and navigation capabilities, thus making these assumptions very unrealistic. Furthermore, computation, energy, and memory limitations impose severe restrictions on the use of advanced map-based navigation algorithms.

In this paper, we propose an original solution based on augmented reality to solve this problem using a simple, almost reactive, algorithm. Light gradients depicted on the arena by a video projector allow the central planner to locally tune the robots' behavior such that it accounts for the plan optimized at the macroscopic level.

More specifically, the central planner draws gradients between spots; intensity at the darker end of the gradient originating from spot i to spot j is related to the net rate of transition from spot i (exit rate $\sum_j k_{ij}$), and the width of the edge is proportional to $k_{ij} / \sum_l k_{il}$. The robot wanders about randomly in the spot until it encounters a gradient at its border. After getting an estimate of $\sum_j k_{ij}$, the robot draws T_{exp} from an exponential distribution with rate $\sum_j k_{ij}$ (or with mean $1/\sum_j k_{ij}$).

If the robot has already spent more than T_{exp} units of time in the spot, it makes the transition. Otherwise, the robot waits until it has spent more than T_{exp} units of time in the spot, and then makes the next transition. This ensures that the rate of exit of robots from the spot is the closest that we can get to $\sum_j k_{ij}$, as required. Also, the probability of choosing an edge to spot j is directly proportional to the relative widths of the edges with respect to one another, which are given by the ratio $k_{ij} / \sum_l k_{il}$. If these choices are made independent of the activity of the robot in

the spot, then by thinning of Poisson processes [12], we know that the transitions from spot i to j is dictated by a Poisson process with the parameter k_{ij}, as required.

The range of intensities that are available to represent the gradients is limited by the operating range of the light sensor as well as the intensity of the projector. Hence, instead of using constraint (4) or (5), a different objective function needs to be *minimized*:

$$f(\mathbf{K}) = \max \sum_j k_{ij} \tag{8}$$

After obtaining the transition matrix $\mathbf{K}$, we can scale it such that the maximum value of $\sum_j k_{ij}$ can be represented by the available intensity range. We denote the scaled transition matrix $\mathbf{K}_{opt}$.

3.2.5 Robot Controller

Finally, one needs to translate the solution described in the previous sections into an actual robot controller. There is no automated way of constructing such a controller; however, in a top-down approach, the designer merely encodes the various requirements prescribed at higher abstraction levels in the robot controller while accounting for the technological limitations of the robotic platform. In our case, robots are programmed with a simple behavior-based controller composed of five states: *search*, *climb*, *in spot*, *probe*, and *u-turn*. Other auxiliary states are used, but they are not mentioned here for the sake of clarity. Figure 2 depicts the state chart of the robot controller.

In the state *search*, the robot performs a simple random walk, i.e., it alternates between forward motion and tumbling in a random direction. If the robot detects a gradient, it transitions to the state *climb*; in this state, the robot moves up the gradient using a simple reactive scheme, very similar to Braitenberg vehicles. When no change in intensity is detected, it means either that the robot is in a spot if the intensity is high (transition to the state *in spot*), or that the robot is lost if the intensity is low (transition to the state *search*). In the state *in spot*, the robot simply moves forward; upon detecting a drop in intensity (i.e., the robot reaches the border of the spot), it transitions to the state *probe* and stops. In the state *probe*, it samples its light sensors for a certain amount of time. Depending on the average of these measurements, the robot may transition to either of two states: (i) if it encountered an outgoing gradient and it has already spent more than T_{exp} units of time in the spot[5], then it transitions to the state *climb*; (ii) if it encountered an incoming gradient or it has spent less than T_{exp} units of time in the spot, then it transitions to the state *u-turn* in order to remain within the spot.

[5] If the robot encounters an outgoing gradient for the first time since entering the spot, it first draws T_{exp} from an exponential distribution whose mean is inversely proportional to the average intensity of the gradient. Furthermore, because of the very limited computational power available on our robotic platform, we use a lookup table to generate these random numbers.

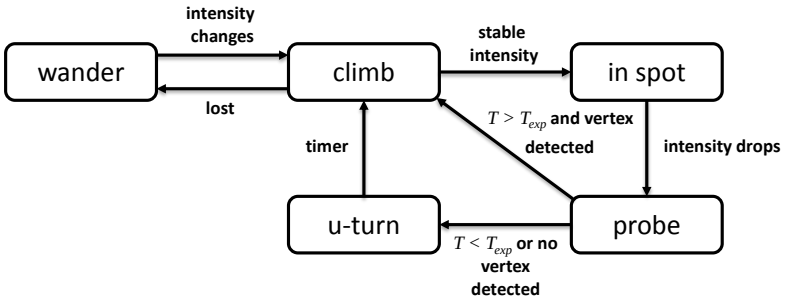


Fig. 2 Finite State Machine (FSM) of the robot controller yielded by the top-down approach.

3.3 Bottom-Up Strategy

In the bottom-up strategy, we start with a very simple behavior-based controller, which we optimize using a suite of models at different abstraction levels. One of the interesting feature of the bottom-up approach is that it deals from the beginning with the intrinsic limitations of the robotic nodes rather than setting requirements that are not necessarily feasible. For instance, in our particular case study, we do not assume that the robots receive broadcasted navigational hints from a central planner, thereby leading to a fully distributed, on-board, more robust control solution. Instead, the robots exploit local information (in our particular case, the intensity of the spots) to tune their behavior in a way that is optimal for a wide range of scenarios (i.e., for various combinations of spot thresholds κ_i for $i = 1,\ldots,N$). More specifically, no gradient is projected on the arena, and the robots move from spot to spot using a random walk similar to that described in Section 3.2.5 instead of a directed gradient-based movement as that used in the top-down approach. Upon detecting a change in light intensity, a robot enters into the spot, and starts exploring it. Each time the robot reaches the spot's border, it will leave with probability $p_{leave}(\hat{\kappa})$, which depends on its estimate $\hat{\kappa}$ of the threshold of the spot. Figure 3(A) depicts the state chart of the robot controller.

One important difficulty with this approach is that we need to define the optimal leaving probabilities $p^{opt}_{leave}(\hat{\kappa})$ for $\hat{\kappa} = 1,\ldots,N_\kappa$ such that the distance between the desired distribution $\mathbf{x}^d$ and the actual distribution $\mathbf{x}(t)$ of robots over the set of spots is, *on average*, minimal. Note that this means that we are only interested in optimizing the average performance over the set of all possible combinations of threshold values.

To solve this problem, we use a genetic algorithm (GA) coupled with a simple macroscopic model of the system to determine the optimal leaving probabilities for a given number of spots and robots. The genome is simply the real valued vector $\mathbf{p_{leave}} = \left[p_{leave}(\hat{\kappa} = 1),\ldots,p_{leave}(\hat{\kappa} = N_\kappa)\right]$, and the population size is 20 individuals (crossover fraction = 0.8, mutation rate = 0.2, with 2 elite children). The fitness function is the square of the l^2-norm (Equation (6)) between the desired distribution $\mathbf{x}^d$ and the actual distribution $\mathbf{x}(t)$ of robots at time t yielded by the set of

leaving probabilities $\mathbf{p_{leave}}$ at $t = 100$ s. Therefore, we do not optimize the system for fastest convergence; rather, we target an optimal, though transient, performance at $t = 100$ s, which is compatible with the dynamics observed in the real system. Also, the optimization does not assume any prior knowledge about the thresholds κ_i of the spots; rather, the fitness of each individual is averaged over a test set of 100 randomly generated scenarios.

However, since a typical GA run involves more than one thousand evaluations of the fitness function, one needs a very efficient computational model for predicting the distribution $\mathbf{x}(t)$. Section 3.3.1 describes the construction of such a model.

3.3.1 Chemical Reaction Network

In this section, we construct the macroscopic model that is used in conjunction with the GA; we use a Chemical Reaction Network (CRN) formalism to model the interactions between robots and spots. A CRN can be represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of vertices $\mathcal{V}$ represents the *complexes*, i.e., state variables that denote the number of robots in a given state. The set of directed edges $\mathcal{E}$ represents the *reactions* between complexes, i.e., the robots' transitions from one state to the other. Each reaction can be denoted by an ordered pair $(i, j) \in \mathcal{V} \times \mathcal{V}$, meaning that one complex i transitions to complex j. Furthermore, each reaction is associated with a rate constant.

From a methodological standpoint, CRNs are well suited to the bottom-up construction of macroscopic models, as one can easily translate the controller of a robot (which is described as a finite state machine) into a CRN (Figure 3).

Therefore, one can write the following set of reactions:

$$X_s \underset{r_{l,i}}{\overset{r_e}{\rightleftharpoons}} X_i \quad \forall i \in \mathcal{S} \tag{9}$$

where $\mathcal{S}$ denotes the set of spots, X_s the number of robots searching for spots, X_i the number robots in spot i, r_e the rate at which a robot encounters a spot i, and $r_{l,i}$ the rate at which a robot leaves the spot it is exploring, which also depends on its threshold κ_i. The identification of the rates r_e and r_l is based on geometric approximations described in previous work [11].

In order to yield quantitative predictions, CRNs are converted into two types of models: (i) macroscopic (macro-discrete) models, i.e., continuous-time Markov processes whose states represent discrete numbers of robots, or (ii) macroscopic continuous (macro-continuous) models of ordinary differential equations (ODE) whose state variables represent continuous fractions of the robotic population.

We use the StochKit toolbox [6] in order to perform stochastic simulations of macro-discrete models. Also, we convert CRNs into macro-continuous models using the relation $\dot{\mathbf{y}} = \mathbf{S} \cdot \mathbf{p}(\mathbf{y})$ where $\mathbf{S} = (s_{ij})$ is the stoichiometry matrix, with the stoichiometric coefficient s_{ij} of the j-th species in the i-th reaction, and is the propensity vector $\mathbf{p}(\mathbf{y})$, which depends on the reaction rates and the population of reactants for each reaction. Then, we numerically integrate this system of ODEs using MATLAB's `ode15s` function. Using this model, the evaluation of one individual

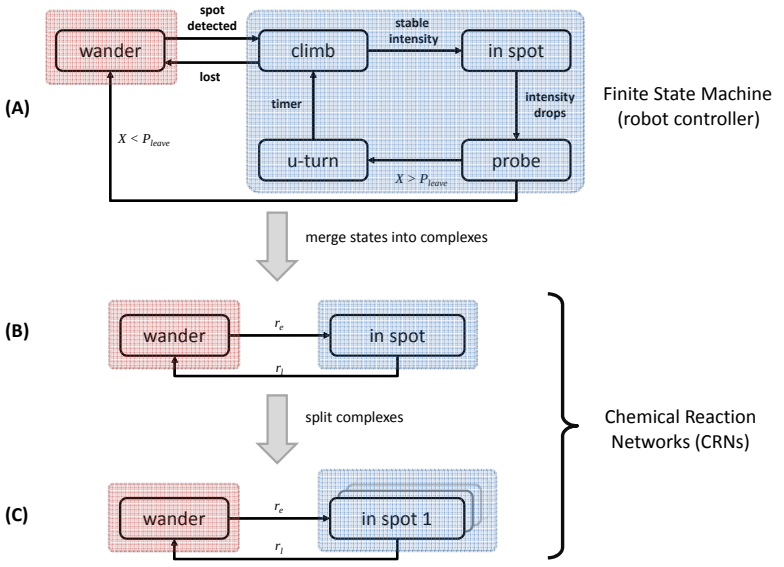


Fig. 3 *Translation of the FSM of the robot's controller into a CRN.* (**A**) → (**B**): The states of the FSM that are irrelevant to the collective dynamics (generally because the robot spends a short amount of time in them, or because they do not modify the state of the ensemble) can be *merged* into a single complex of the CRN (e.g., the states *climb*, *in spot*, *u-turn*, and *probe* are merged into a single complex *in spot*). (**B**) → (**C**): The complexes that need to account for states that are hidden at the level of the robot's controller can be split into multiple sub-complexes (e.g., the model needs to track the spot the robot is in, and the complex *in spot* is therefore split into N sub-complexes).

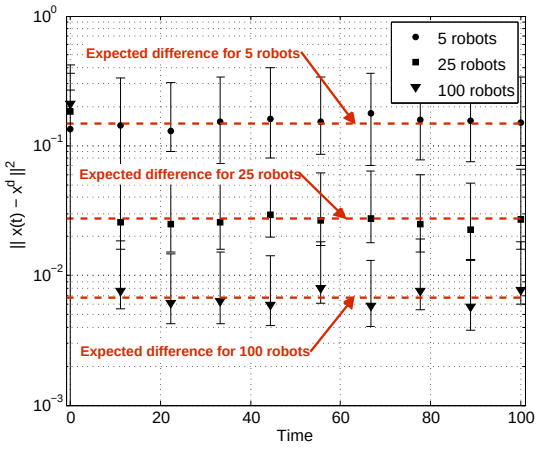


Fig. 4 Distance (on a logarithmic scale) between the actual distribution at time t yielded by stochastic simulations of the top-down approach for different number of robots (30 runs, marker denotes the mean, and error bars the 95% confidence interval) and the desired distribution $\mathbf{x}^d$. The dashed horizontal lines denote the expected distance according to Equation (7). We note a strong agreement between our numerical simulations and the analytical prediction.

takes about 3.5 seconds on 2.66 GHz quad-core processor with 8 GB of RAM, which is two orders of magnitude faster than 100 runs of stochastic simulations using StochKit.

Table 2 Summarized results of three experiments (top-down, bottom-up, and baseline, 10 runs of 20 minutes each) using 6 real Alice robots and 4 spots (with $\kappa \in [1,2,3]$). The system performance is computed using Equation (1). The distance between the actual distribution $\mathbf{x}(t)$ and the desired distribution $\mathbf{x}^d$ of robots is given by Equation (6), and averaged over all sampled data points.

Metric	**Moment**	**Case studies**		
		Top-down	*Bottom-up*	*Baseline*
Performance	*mean*	0.81	0.89	0.58
	median	0.85	0.95	0.57
	std dev	0.25	0.27	0.22
Distance $\|\mathbf{x}(t)-\mathbf{x}^d\|^2$	*mean*	0.36	0.37	0.25
	median	0.39	0.36	0.25
	std dev	0.11	0.06	0.02

4 Results and Discussion

As discussed in Section 3.2.2, the top-down approach relies on an optimization scheme that assumes an infinite number of robots[6]; therefore, the actual distribution of robots over the set of spots is always different, on average, from the optimal distribution. The distance between these two distributions can be predicted for any given number of robots using Equation (7). Figure 4 provides a validation[7] of this expected distance using stochastic simulations.

One of the main findings of our experimental study is that the top-down approach does not perform significantly better (or worse) than the bottom-up approach with the leaving probabilities yielded by the GA (i.e., $\mathbf{p}^{\mathbf{opt}}_{\mathbf{leave}} = \big[p_{leave}(\hat{\kappa}=1),\ldots,p_{leave}(\hat{\kappa}=N_\kappa=3)\big] = \big[0.335, 0.0002, 0.005\big]$). However, one also needs to check that these optimized controllers do not actually perform the same as any other non-trivial controller. We rule out this hypothesis by performing a third type of experiments that use a so-called *baseline* controller, which is essentially the bottom-up controller with all leaving probabilities set to zero. A non-parametric statistical test shows that both the top-down and the bottom-up controllers perform *significantly* better (with respective p-values of 0.037 and 0.025 using Mann-Whitney test) than the baseline controller. Table 2 summarizes the results of these experiments. In particular, we shall outline that the large variability of both the system performance and the average distance between the actual distribution and the desired distribution of robots does not allow one to conclusively determine which of the optimized controllers perform best in the context of study. Our results merely show that a sophisticated approach such as the one proposed by Berman et al. [1] does not bring any significant performance increase in the context of our case study, in spite of the supplementary infrastructure it uses.

[6] Note that the bottom-up approach proposed in this paper also assumes an infinite number of robots for optimizing the system. However, this assumption is a deliberate choice of the designer made for efficiency purposes (since it is a sufficient condition that allows for converting the CRN to a macro-continuous model, which is several orders of magnitude faster than stochastic simulations), and supported by a careful validation of the macroscopic models.

[7] This result is validated here in the context of the top-down approach, but it is very general and valid for the bottom-up approach as well.

There are multiple reasons why the extra information provided to the robots in the top-down approach does not significantly improve the system's performance. First, when the density of spots is high enough, a simple search behavior such as the one used in the bottom-up approach does not perform significantly worse than gradient ascent, because robots can compensate for their lack of information with higher velocities. Second, the interaction between robots in a spot dramatically affects the transition rates; for instance, when two robots collide and avoid each other near the border of a spot, one may get lost during the maneuver. Also, when a spot is explored by many robots, the time they spend avoiding each other becomes non-negligible, thus modifying the effective exit rate. Last, the assumptions underlying our formulation of the expected system performance provided in Section 3.1.1 are not necessarily met in reality. In particular, dispatching more than κ_i robots to a given spot i may actually be beneficial because, again, it makes the system more robust; this fact might explain the discrepancy between performance and distance to the desired distribution observed in our experimental results. Finally, and most importantly, the presence of noise in sensor measurements and the heterogeneity of the projected picture dramatically affect the effective transition rates.

As a result, robust approaches are favored over complex strategies. In our bottom-up approach, leaving probabilities are not specifically optimized for a given configuration, but rather for a wide range of scenarios; therefore, the performance loss observed during the process of implementation is less than in the case of the system developed using a top-down scheme. To verify this assertion, we performed 50 runs of realistic simulations using Webots, a physics-based mobile robot simulator. On the one hand, the top-down approach performed significantly better in simulation (mean = 2.21, median = 2.21, std dev = 0.35) than in reality (mean = 0.81, median = 0.85, std dev = 0.25, see Table 2). However, the performance of the bottom-up approach in simulation remained stable (mean = 0.91, median = 0.95, std dev = 0.35). The main difference between these simulations and reality lies in the modeling of noise; we assume a Gaussian noise on the light sensors, but the heterogeneity of the projected picture actually yields time-dependent and space-dependent multimodal noise distributions in reality. Also, most of the technological limitations of the platform (e.g., memory, resolution of analog/digital conversion, limited floating point handling) are not captured in the simulations.

These findings confirm the better robustness of the bottom-up approach; however, the top-down approach allows for a much better theoretical tractability at the macroscopic level, and, in particular, the use of efficient optimization methods such as linear and non-linear convex optimization.

5 Conclusion

In this paper, we compared the strengths and weaknesses of bottom-up and top-down model-based methodologies for designing distributed controllers. Of course, our study does not expose all the features that make either approaches more or less suitable to a given system; this endeavor is by itself a whole body of future research. The main claim of this paper is that conventional top-down design of multi-

robot systems is generally not amenable to efficient implementation using resource-constrained robots; faithful and computationally efficient models built incrementally from the bottom up are an essential ingredient to the design and the control of such systems. Ultimately, we believe that these two approaches should be combined into a more powerful model-based control design methodology that has the potential to achieve higher, more tunable coordination at the macroscopic level while incorporating all a priori known technological limitations at the microscopic level.

Acknowledgements The authors would like to thank Emmanuel Droz for the development of the extension board for the Alice robot. This work is partially sponsored by the Swiss research initiative Nano-Tera.ch.

References

1. S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Trans. On Robotics*, 25(4):927–937, Aug. 2009.
2. G. Caprari, T. Estier, and R. Siegwart. Fascination of down scaling - Alice the sugar cube robot. *J. of Micromechatronics*, 1(3):177–190, 2002.
3. R. Fletcher. Semi-definite matrix constraints in optimization. *SIAM J. On Control and Optimization*, 23(4):493–513, 1985.
4. M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx, May 2010.
5. A. Ijspeert, A. Martinoli, A. Billard, and L. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, Sept. 2001.
6. H. Li, Y. Cao, L. R. Petzold, and D. T. Gillespie. Algorithms and software for stochastic simulation of biochemical reacting systems. *Biotechnology Progress*, 24(1):56–61, Jan 2008.
7. T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli. Swistrack - a flexible open source tracking software for multi-agent systems. In *Proc. of the 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2008)*, pages 4004–4010, 2008.
8. A. Martinoli, K. Easton, and W. Agassounon. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *Int. J. Robotics Research*, 23(4-5):415–436, Jan 2004.
9. L. Matthey, S. Berman, and V. Kumar. Stochastic strategies for a swarm robotic assembly system. In *Proc. of the 2009 IEEE Int. Conf. on Robotics and Automation (ICRA 2009)*, pages 1953–1958, May 2009.
10. G. Mermoud, J. Brugger, and A. Martinoli. Towards multi-level modeling of self-assembling intelligent micro-systems. *Proc. of the 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, 1:89–96, May 2009.
11. G. Mermoud, L. Matthey, W. Evans, and A. Martinoli. Aggregation-mediated collective perception and action in a swarm of miniature robots. In M. Luck, S. Sen, W. van der Hoewk, and G. Kaminka, editors, *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 599–606, Toronto, Canada, May 2010.
12. S. M. Ross. *Introduction to Probability Models, Ninth Edition*. Academic Press, Inc., Orlando, FL, USA, 2006.
13. A. Winfield, W. Liu, J. Nembrini, and A. Martinoli. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2(2):241–266, 2008.